9) Decompression and unpacking of all fetch objects into application work area, and logs completion status.

10) Returns control to the application (or optional transporter user interface).

The product checks the completion code, and completed object manifest to deal with any error conditions. The application performs any required import processing on fetched objects to integrate the data and indexes with prior data, as desired, to enable seamless use. If desired, import processing can include, or offer as a user selection, file maintenance functions relevant to the information product including, for example, file purging to remove obsolete information files and preserve the user's storage space. Specifications of files to be deleted can be included with the original product or with a fetch object. In either event the responsibility for accurate specification is passed to the vendor, relieving the user of the risk of making erroneous deletions and anxiety attendant thereon. After such import processing the containing information product (or the optional separate user interface) then returns control to the user for use of the received data.

Those skilled in the art will appreciate that the identification of files in the object manifest, or for file maintenance functions as the user station, or for any other purpose of the invention, can be effected generically, for example by using wild card characters, as is customary in file specification, and which effectively permits multiple objects to be specified as a class related by file name characteristics, or related individually, thereby providing options for specifying of such class of multiple objects to proceed at one time or in a series of transports over time. Other algebraical identification methods can be used which may reference object versions in series or comparable characteristics.

The foregoing steps are illustrated in the flow block diagram of Figure 2. When containing information product 12 issues an information transport call 51, setup filter 52 runs setup routine 54 if this is a first call and no information transport setup was run on installation of containing information product 12. At block 56, an object manifest is retrieved for pre-transport preparation at

APPENDIX B

has already added ones for Compact Pro and StuffIt and the like, and I'm sure Dantz will be modifying that Compress selector in the future so it includes AutoDoubler/DiskDoubler by default as well.

Let's explain what just happened and what it will do. You modified the Compress selector so that it knows NOT to compress files created by AutoDoubler or DiskDoubler (or any of the other common compression programs). Now there's no problem keeping compression on in **Retrospect** because **Retrospect** will look at each file to backup and see if it was created by a compression program. If that's true (i.e. the file matches the Compress selector), then **Retrospect** will simply backup the file and not compress it. If the file is not compressed by one of the compression programs, then **Retrospect** will use its own compression, saving space on the backup disks. This sort of stuff may not be all that easy, but at least it's possible. Hats off to Dantz for providing a program with this level of power and flexibility so it can handle strange situations like this.

Retrospect Remote

One confusing issue which has arisen on the Internet is what happens with something like **Retrospect Remote**? This is an extension (INIT) which resides on networked Macs and acts as a file handler for **Retrospect** running on a "backup server" on the network. When **Retrospect** wishes to backup John's Mac, it sends a message to the **Retrospect Remote** extension on John's Mac, which scans John's disk and hands back a list of files. **Retrospect** (on the backup server) decides what needs to be backed up and then asks the **Retrospect Remote** on John's Mac to start sending it files over the network.

These files will come across the network expanded. This is because **Retrospect Remote** uses standard toolbox calls to copy the files; AutoDoubler intercepts the calls (as usual) and expands the files. This is the opposite of how it works on your local Mac, which receives the files literally. Why the difference? Because there's no easy way for an extension (like **Retrospect Remote**) to indicate to another extension (like AutoDoubler) that the extension is asking for a file; on the other hand, it's easy for AutoDoubler to tell that, for example, **Retrospect** is asking for a file on a local Mac. On a networked Mac running **Retrospect Remote**, all AutoDoubler knows is that _something_ is asking for a file and, if that file is compressed, it needs to be expanded. It appears that there is no workaround - you'll just have to leave **Retrospect** compression on or else all your files will be in your backup in expanded form and will take up twice as much space. Lars Holm of Dantz Tech Support said that they were testing all the compression programs to ensure compatibility and that Dantz is looking for ways to let other companies detect the presence of the **Retrospect Remote** extension so that files do not have to be expanded all the time.

Dantz Development -- 510/849-0372
Alysis -- 415/566-2263
Salient -- 415/321-5375
Aladdin -- 408/761-6200

Information from:
Lars Holm, Dantz Development-- DANTZ.TECH@applelink.apple.com
Lloyd Chambers, Salient -- salient@aol.com